อาจหาญ สัตยารักษ์

SOFT MAIL

การลบไฟล์ *.BAK ในฮาร์ดดิสค์

แนวคิดที่ได้จากโปรแกรมนี้ หากรู้จักคิดก็จะสามารถปรับปรุงให้ตรงกับความต้องการของตนเองได้อีกด้วย

ปัญหาของโปรแกรมเมอร์หรือผู้ใช้เวิร์ดบนดอสอย่างหนึ่งคือ การที่ระบบงานที่ตนเองใช้นั้นได้สร้างไฟล์สำเนาขึ้นมาให้โดยอัตโนมัติ โดยใช้ชื่อเดิม แต่ให้ส่วนขยายเป็น BAK เมื่อเขียนโปรแกรมนานวันเข้า ย่อมมีเหล่านี้มากมายในฮาร์ดดิสค์ และแอบเร้นอยู่ในหลาย ๆ พาธ

การที่จะลบไฟล์ *.bak ที่แยกกันอยู่หลาย ๆ พาธนั้นเป็นเรื่องที่ใช้เวลาพอสมควร เพราะคำสั่งในดอสที่คนนิยมใช้ตามปกติคือ del หรือ erase นั้นมีการใช้งานดังนี้

```
DEL [drive:][path]filename [/P]
ERASE [drive:][path]filename [/P]
```

ซึ่งตามรูปแบบนี้จะต้องสั่งลบทีละพาธเท่านั้น ไม่สามารถใช้คำสั่งเดียวลบทั้งฮาร์ดดิสค์ได้

เขียนไฟล์เองดีกว่า

ในฐานะโปรแกรมเมอร์ การเขียนไฟล์ใช้เองจึงเป็นเรื่องที่ทำกันง่าย ๆ รวมทั้งสามารถพัฒนาให้ทำงานได้เก่งกว่าดอสทั้งหลายอีกด้วย ซึ่งผู้เขียนจะนำมากล่าวถึงในภายหลัง

การเขียนโปรแกรมฉบับนี้ผู้อ่านจะได้แนวคิดต่าง ๆ ดังนี้

1. การตรวจสอบสถานะของไฟล์
2. วิธีการลบไฟล์ *.bak ที่ค้นพบ
3. วิธีการตรวจสอบไฟล์ทั้งฮาร์ดดิสค์
4. ได้รับรู้ถึงขีดความสามารถของโปรแกรมแบบที่เรียกตัวเองซ้ำกันหลาย ๆ ชั้นดียิ่งขึ้น

สำหรับข้อสี่นั้น ต้องสังเกตความสามารถในการจดจำลำดับของไฟล์ พาธที่เปลี่ยนไป (ซึ่งอาจมีหลายชั้นด้วยกัน)

ไฟล์ใช้งานนี้มีเพียง 78 บรรทัดเท่านั้น

สำหรับซอฟต์เมล์จะได้ไฟล์ del_bak.pas และ del_bak.exe รวม 2 ไฟล์

ลบไฟล์กันเลย

เราจะเริ่มต้นด้วยการกำหนดตัวแปรและชุดคำสั่งลบไฟล์กันเป็นอันดับแรก ดังรูปที่ 1 คำอธิบายมีดังนี้

1. บรรทัดที่ 1-4 กำหนดตัวแปรเริ่มต้นเพื่อเก็บจำนวนไฟล์และพาธที่ค้นพบ จำนวนพาธ จำนวนไฟล์ที่ลบ (*.bak)

2. บรรทัดที่ 5-12 ส่วนการลบไฟล์ที่ตรวจสอบพบ โดยเริ่มด้วยการแสดงชื่อไฟล์บนจอภาพ กำหนดการใช้ไฟล์นั้นแล้วทำการลบไฟล์ โปรดจำไว้ว่าไฟล์ที่จะลบนี้จะต้องไม่ใช่ไฟล์ที่เปิดใช้งานอยู่ กล่าวคือ ต้องไม่มีการกำหนดคำสั่ง reset(f,1); รวมทั้งไม่ต้องปิดไฟล์ close(f); เนื่องจากไฟล์นี้ไม่ได้ถูกเปิดใช้งาน (ถูกลบไปแล้วด้วย) พร้อมกับนับจำนวนไฟล์ที่ถูกลบ

```
1: PROGRAM Del_backUpFile;
2: uses crt,dos;
3: var countOffind,CountOffile,CountOfDir,CountOfBak:longint;
4:     PathOld,DriveDel:string;

5: PROCEDURE EraseFile(FileRead:string);
6: var f:file;
7: begin
8:   textcolor(11); writeln(Fileread);
9:   assign(f,fileread);
10:   $I- Erase(f);  $I+
11:   inc(CountOfBak);
12: end;
```

รูปที่ 1
ตัวแปรเริ่มต้น
และชุดคำสั่งลบไฟล์

```
13: PROCEDURE Find_File_BAK;
14: var filename,PathOld,PathOfsource,st:string;
15:     dirInfo:SearchRec;  Path:boolean;
16: label Arthan;
17: begin
18:     getdir(0,Pathofsource); Pathold:=Pathofsource;
19:     if Pathofsource[0]>#3 then Pathofsource:=Pathofsource+'_plain
20:     findfirst('*.*',$3F,dirInfo);
21:    while DosError = 0 do
22:     begin
23:       path:=false;
24:       if DirInfo.attr and 16=16 then path:=true;
25:       if DirInfo.attr in [40] then goto Arthan;
26:       filename:=DirInfo.Name;
27:       if filename='.' then  goto arthan;
28:       if filename='..' then  goto arthan;
29:       inc(countOffind); textcolor(10);  st:='';
30:       if path then
31:        begin
32:         textcolor(13); st:=' <DIR>'; inc(Countofdir);
33:        end
34:         else inc(Countoffile);
35:       writeln(Pathofsource+filename+st);
36:    if path   then
37:       begin
38:         $I- chdir(Pathofsource+filename); $I+
39:          if Ioresult=0 then Find_File_BAK;
40:          chdir(Pathold);
41:          getdir(0,Pathofsource); Pathold:=Pathofsource;
42:          if Pathofsource[0]>#3 then Pathofsource:=Pathofsource+'_plain
43:       end
44:         else
45:            if Pos('.BAK',PathOfsource+Filename)<>0 then
46:                                  EraseFile(PathOfsource+Filename);
47:  arthan:
48:       findnext(dirInfo);
49:     end;
50: end;
```

รูปที่ 2
ชุดคำสั่งแบบเรียกตนเอง

ไฟล์เรียกตนเอง

สิ่งสำคัญของการลบไฟล์ทั้งฮาร์ดดิสก์ คือ การที่ต้องเปลี่ยนพาธทำงานไปยังพาธที่ตรวจสอบพบ ซึ่งอาจจะมีซ้อนกันหลาย ๆ ชั้น ขอแทรกถามปัญหาที่ผู้เขียนยังไม่เฉลยในตอนนี้ไว้ว่า "ใครทราบบ้างว่าเราสามารถสร้างพาธซ้อน ๆ กันได้สูงสุดกี่ชั้น และเงื่อนไขอะไรเป็นตัวกำหนดจำนวนชั้นของพาธย่อย ๆ นั้น" โดยขอใบ้ว่าเป็นเรื่องของความยาว

ความสามารถอย่างหนึ่งของภาษาปาสคาลคือ การเขียนโปรแกรมแบบเรียกตนเอง ซึ่งจะมีการกำหนด ตัวชี้ (pointer) ขึ้นเองโดยอัตโนมัติ โดยโปรแกรมเมอร์ไม่จำเป็นต้องกำหนดอะไรเพิ่มเติมเลย

การทำงานที่เรากำหนดไว้จะมีดังนี้

1. ตรวจสอบไฟล์ทั้งหมดในฮาร์ดดิสก์
2. แยกไฟล์ที่ตรวจสอบได้เป็น 4 กลุ่ม คือ

2.1 ไฟล์ปกติ

2.2 ชื่อพาธ เมื่อพบเราจะต้องเปลี่ยนพาธ

2.3 ชื่อเลเบล

2.4 ไฟล์เริ่มต้นของพาธ (ใช้เป็นตัวชี้ คือไฟล์ที่เรามองเห็นเมื่อใช้คำสั่ง dir จะเห็นเป็น

. <DIR>

.. <DIR>

รายการที่ 2.3 และ 2.4 เราจะข้าม ไม่นำมาดำเนินการใด ๆ

3. ถ้าพบไฟล์ *.BAK เราก็จะทำการลบไฟล์นั้น ๆ ออกไปจากฮาร์ดดิสค์

ชุดคำสั่งแบบเรียกตนเองดังรูปที่ 2 คำอธิบายของโปรแกรมมีดังนี้

1. บรรทัดที่ 14-16 กำหนดตัวแปรใช้งาน

2. บรรทัดที่ 18-19 จดจำว่าพาธใช้งานขณะเริ่มต้นชุดคำสั่ง เพื่อนำไปใช้ในการลบไฟล์

3. บรรทัดที่ 20 เริ่มต้นค้นหาชื่อไฟล์และพาธ โดยกำหนดเป็น $3F ซึ่งตามรูปแบบที่ภาษาปาสคาลได้กำหนดไว้เป็นดังนี้

$01 = ReadOnly

$02 = Hidden

$04 = SysFile

$08 = VolumeID

$10 = Directory

$20 = Archive

$3F = AnyFile

4. บรรทัดที่ 21-49 ส่วนของการทำงานทั้งหมด

5. บรรทัดที่ 23-24 ส่วนการตรวจสอบว่าเป็นพาธ (Directory) หรือไม่ เพราะถ้าเป็นเราจะต้องทำการเปลี่ยนพาธไปยังพาธที่ตรวจพบ

6. บรรทัดที่ 25 ตรวจว่าเป็นชื่อของเลเบลหรือไม่ ถ้าใช่จะข้ามไปบรรทัดที่ 47 ถ้าไม่ใช่ก็จะตรวจสอบตามคำสั่งบรรทัดถัดมา

7. บรรทัดที่ 26-28 ตรวจสอบชื่อไฟล์ ถ้าพบว่าเป็นแค่ตัวชี้ คือชื่อไฟล์เป็น . หรือ .. จะข้ามไปยังบรรทัดที่ 47 เช่นกัน

8. บรรทัดที่ 29-35 นับจำนวนไฟล์และพาธที่ค้นพบ และแสดงชื่อที่ค้นพบบนจอภาพ โดยแสดงข้อความ <DIR> ต่อท้ายชื่อที่เป็นพาธด้วย

9. บรรทัดที่ 36-43 ถ้าพบว่าส่วนนี้เป็นพาธ ก็จะทำการเปลี่ยนพาธและเรียกชุดคำสั่งนี้ซ้ำ (ส่วนนี้คือการเรียกตนเอง) ตามบรรทัดที่ 39 เมื่อทำงานเรียกตนเองเสร็จสิ้น จะย้อนกลับคืนพาธเดิมตามคำสั่งบรรทัดที่ 40-42

10. บรรทัดที่ 45-46 หากเป็นชื่อไฟล์และมีส่วนขยายเป็น .BAK ก็จะทำการลบไฟล์ ดังนั้นตรงนี้หากท่านต้องการลบไฟล์ที่มีส่วนขยายอื่น ๆ ก็สามารถกำหนดได้เช่นกัน หรืออาจกำหนดให้ลบได้หลายส่วนขยายก็ได้

11. บรรทัดที่ 48 คำสั่งค้นหาชื่อไฟล์หรือพาธที่หลงเหลืออยู่ หากหมดชื่อไฟล์หรือพาธก็จบการทำงานตามเงื่อนไข while DosError = 0 ในบรรทัดที่ 21

โปรดสังเกตว่าคำสั่งชุดนี้สั้นมาก แต่เมื่อทำงานจะสามารถเปลี่ยนพาธลงไปได้ลึกสุดถึงชั้นสุดท้ายของแต่ละพาธได้ โดยที่โปรแกรมเมอร์ไม่ต้องสนใจว่าจะมีพาธซ้อนกันอยู่เท่าใด

จำนวนพาธที่จะอยู่ซ้อนกันได้นั้น จะมีกี่ชั้นขึ้นอยู่กับความยาวของชื่อพาธด้วย เพราะสุดท้ายนั้นดอสได้กำหนดว่าเมื่อลงไปลึก ๆ แล้วความยาวของชื่อพาธนั้นจะต้องป้อนได้ในหนึ่งบรรทัดของจอภาพเท่านั้น มาถึงตอนนี้คงมีผู้อ่านนึกออกบ้างแล้วว่าจะสร้างพาธซ้อนลงกันไปได้นั้นขึ้นกับเงื่อนไขอะไร

ส่วนของโปรแกรมหลัก

หลังจากนี้ก็มาถึงการนำเอาชุดคำสั่งทั้งหมดมาทำงานเสียที โดยผู้เขียนมีแนวในการเขียนโปรแกรมหลักดังนี้

1. ลบจอและแสดงข้อความที่บรรทัดแรกว่าเป็นโปรแกรมอะไร เขียนโดยใคร

2. กำหนดขอบเขตของจอภาพให้อยู่เฉพาะบรรทัดที่ 2 ลงมา ทั้งนี้เพื่อให้ข้อความที่เราแสดงในข้อ 1 นั้นยังคงปรากฏบนจอภาพ แม้ว่าจะมีการแสดงชื่อไฟล์หรือพาธมากกว่า 24 ไฟล์ (การแสดงข้อความมาก ๆ นั้นจะมีการเลื่อนข้อความขึ้นข้างบนไปเรื่อย ๆ ถ้าเราไม่กำหนดขอบเขตจอภาพไว้ บรรทัดแรกที่แสดงก็จะหายไปเช่นกัน)

3. เรียกใช้ชุดคำสั่ง ซึ่งจะมีการแสดงชื่อไฟล์และพาธไปเรื่อย ๆ

4. เมื่อจบการตรวจสอบที่ฮาร์ดดิสค์แล้ว ก็จะให้แสดงว่า พบไฟล์และพาธจำนวนเท่าใด เป็นพาธเท่าใด และได้ลบไฟล์ *.bak ไปกี่ไฟล์

5. ยกเลิกการกำหนดขอบเขตจอภาพ

นำแนวคิดนี้มาเขียนโปรแกรมได้ดังรูปที่ 3 มาดูคำอธิบายโปรแกรมดังนี้

1. บรรทัดที่ 53-55 ลบจอภาพ และกำหนดค่าให้ตัวแปรใช้งาน

2. บรรทัดที่ 56-60 ตรวจสอบว่ารันที่พาธใด และตรวจสอบว่าได้กำหนดพารามิเตอร์ตอนรันไฟล์นี้หรือไม่ ถ้ากำหนดก็จะได้ค่าตัวแปร DriveDel เป็นชื่อไดรฟ์ที่กำหนด

ตัวอย่างการกำหนดพารามิเตอร์จะเป็นดังนี้

C:\>del_bak ลบไฟล์ *.bak ในไดรฟ์ C

C:\>del_bak d: ลบไฟล์ *.bak ในไดรฟ์ D

โดยจะมีการเปลี่ยนไดรฟ์ไปตามที่กำหนดในบรรทัดที่ 57-60

3. บรรทัดที่ 61-62 แสดงข้อความลงในบรรทัดแรก

4. บรรทัดที่ 63 กำหนดขอบเขตของจอภาพที่จะใช้ต่อไปนี้ว่าเป็นบรรทัดที่ 2-25 เท่านั้น

5. บรรทัดที่ 65 เรียกใช้ชุดคำสั่งแบบเรียกตนเอง

```
51: ============================DEL_BAK.PAS ========================
52: begin
53:     textcolor(7); textbackground(0);  clrscr;
54:     Countoffind:=0;  Countoffile:=0;
55:     Countofdir:=0;  countOfbak:=0;
56:     getdir(0,pathold);  DriveDel:=paramstr(1);
57:     $I-
58:     if DriveDel='' then chdir(Pathold[1]+':_plain
59:                   else  chdir(DriveDel[1]+':_
60:     $I+
61:     textcolor(15); textbackground(1);
62:     writeln('          Delete File  *.BAK  in HARD DISK          ');
63:     window(1,2,80,25);
64:     textcolor(11); textbackground(0);
65:     Find_File_BAK;
66:     textcolor(14); textbackground(0);
67:     writeln('************************************************');
68:     textcolor(11); textbackground(0);
69:     writeln('Max    ',Countoffind:10,' files & dirs');
70:     writeln('       ',Countoffile:10,' files');
71:     writeln('       ',Countofdir:10,' dirs');
72:     writeln('Erase ',Countofbak:10,' files (*.BAK)');
73:     textcolor(7); textbackground(0);
74:     writeln('         ');
75:    $I-    chdir(Pathold); $I+
76:     writeln; writeln;
77:     window(1,1,80,25); gotoxy(1,22);
78: end.
```

รูปที่ 3 ส่วนของโปรแกรมหลัก

6. บรรทัดที่ 66-74 แสดงจำนวนไฟล์ พาธ และไฟล์ที่ถูกลบไป

7. บรรทัดที่ 75 เปลี่ยนพาธมายังพาธที่รันไฟล์นี้

8. บรรทัดที่ 77 คืนขอบเขตจอภาพให้คงเดิม

หวังว่าผู้อ่านคงได้แนวคิดจากโปรแกรมนี้บ้าง และหากรู้จักคิดก็จะสามารถปรับปรุงให้ตรงกับความต้องการของตนเองได้อีกด้วย

แม้ว่าในปัจจุบันจะมีคนบอกว่าการทำงานบน Windows 95 (หรือ 97) สบายกว่า แต่.. ผู้เขียนยังเชื่อ มีปัญหาบางประการที่ Windows ไม่สามารถช่วยเราได้เช่นกัน ตัวอย่างง่าย ๆ คือให้ลองทำดังต่อไปนี้

1. ในเครื่องคอมพิวเตอร์มีฮาร์ดดิสค์ 2 ตัว โดยไดรฟ์ C เป็น Windows 95 ส่วนไดรฟ์ D นั้นไม่มีอะไรเลย

2. เข้าสู่ Windows 95

3. ลองสำเนาข้อมูลจากฮาร์ดดิสค์ C ไปยังฮาร์ดดิสค์ D (สำเนาทั้งหมดฮาร์ดดิสค์เลย) โดยผ่านขบวนการของ Windows 95

ผู้อ่านลองทำดูว่ามีปัญหาอะไรเกิดขึ้น

คงมีแค่นี้ครับ พบกันใหม่ในฉบับต่อไป